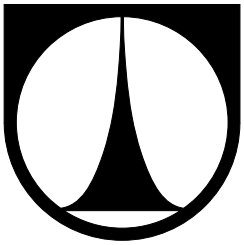
TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií



Samořízené auto na autodráhu

Bakalářská práce

Liberec 2012 Jan Šimon

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B 2612 – Elektrotechnika a informatika

Obor: 2612R011 – Elektronické informační a řídicí systémy

Samořízené auto na autodráhu

Self-driven slot car

Bakalářská práce

Autor práce: **Jan Šimon**

Vedoucí práce: Ing. Jan Koprnický, Ph.D.

Konzultant práce: –

V Liberci 16. května 2012

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Šimon**
Osobní číslo:	**M09000086**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronické informační a řídicí systémy**
Název tématu:	**Samořízené auto na autodráhu**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s částmi elektronického systému automaticky řízeného elektrického auta na autodráhu.
2. Realizujte hardwarovou část systému.
3. Navrhněte algoritmus řízení a vytvořte odpovídající software.
4. Funkční auto otestujte na závodní autodráze.

Rozsah grafických prací:	**dle potřeby dokumentace**
Rozsah pracovní zprávy:	**cca 40–50 stran**
Forma zpracování bakalářské práce:	**tištěná/elektronická**

Seznam odborné literatury:

[1] **Brejl, M.; Necesany, J.: Student's contest: Self-driven slot car racing. Oct. 2008, ISBN 978-83 60810-14-9, ISSN 1896-7094**

[2] **Ďaďo, S.; Kreidel, M.: Senzory a měřicí obvody. Praha : ČVUT, druhé vydání, 1999, ISBN 80-01-02057-6.**

[3] **Palkovič, L.: Autonómne riadené autíčko pre autodráhu. Automatizácia a regulácia URPI ŠVOČ 2009, 2009.**

Vedoucí bakalářské práce:	**Ing. Jan Koprnický, Ph.D.** Ústav mechatroniky a technické informatiky
Datum zadání bakalářské práce:	**14. října 2011**
Termín odevzdání bakalářské práce:	**18. května 2012**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

doc. Ing. Petr Tůma, CSc.
vedoucí ústavu

V Liberci dne 14. října 2011

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/ 2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Mladá Boleslav, duben 2012

Jan Šimon

Poděkování

Děkuji Ing. Janu Koprnickému, Ph.D. za cenné rady a odborné vedení při zpracování této bakalářské práce. Dále patří moje poděkování rodičům za jejich trpělivost a podporu při studiu.

Abstrakt

Bakalářská práce se zabývá tvorbou samořídicího auta na autodráhu. Základem auta je procesor. Model auta obsahuje optické senzory, senzor zrychlení, měření napětí a otáček. Při realizaci jsou použity tyto programy CodeWarrior, Matlab Virtual box XP a TeXnicCenter.

Klíčová slova

Auto, autodráha, procesor, samořízené, bootloader, SD karta, optický senzor, záložní zdroj, freescale, pulsně šířková modulace, akcelerometr, Kalmanova filtrace.

Abstract

My bachelor thesis contains creation of self driven slot car. Main heart of car is processor. Car Model includes optical sensors, accelerometer, measuring voltage and speed. For realisation are used these programs CodeWarrior, Matlab Virtual XP box and TeXnicCenter.

Key words

Car, track, processor, self-driven, bootloader, SD card, optical sensor, standby power supply, freescale, pulse width modulation, accelerometer, Kalman filtering.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Tématem této bakalářské práce je realizovat samořídicí systém pro auto na autodráhu. S tímto autem se zúčastním soutěže Freescale Race Challenge 2012.

Do závodů se přihlásilo sedm týmů z Technické univerzity v Liberci. Vítěz této soutěže postoupí do celostátního kola. Každé auto pojede dva závody na známé trati a dva na trati neznámé. Závod se skládá z osmi kol. Při výpadku z dráhy smí účastník vrátit auto na trať v místě výpadku.

Auto se tedy musí pohybovat takovou rychlostí, aby se nedostalo mimo dráhu – hlavně v zatáčkách. Zároveň musí dosáhnout co nejvyšší rychlosti. Proto je nutné rychlost řídit pomocí systému.

V úvahu připadají dvě možnosti řízení:

- pomocí regulace v reálném čase,
- pomocí predikce tratě z uložené dráhy.

Regulace v reálném čase je složitá pro tuto úlohu vzhledem k práci s velmi krátkými časovými úseky. Druhá možnost se jeví výhodněji pro konstrukci auta a celkovou realizaci. Nejprve autíčko projede dráhu konstantní rychlostí a pomocí senzorů uloží mapu dráhy.
V programu se vyhodnotí rychlosti pro jednotlivé úseky trati a časy, kdy se mají rychlosti změnit.

Centrální součástkou tohoto systému je procesor MCF51JM128, který bude přijímat signály z různých senzorů například akcelerometr, senzor rychlosti, detekce startu atd. Procesor bude pomocí pulsně šířkové modulace vysílat požadavky do H-můstku MC33931, který řídí rychlost a směr stejnosměrného motorku.

V průběhu konstrukce je nutné vyřešit řadu dílčích problémů. Pro příklad zde uvedu některé z nich:

- auto se po výpadku z tratě musí opět sesynchronizovat s uloženou trasou,
- auto po vypadnutí z dráhy musí spustit záložní zdroj,
- ukládání tratě na paměťové médium v našem případě SD kartu,

- jak správně filtrovat data z akcelerometru.

Tuto práci jsem si vybral, protože se mi zdá zajímavé zkombinovat praktické a programovací schopnosti zábavnou a soutěživou formou s ostatními týmy.

Zároveň si osvojím programování procesoru v prostředí CodeWarrior a navrhnout využití různých senzorů. Chtěl bych se o této problematice dozvědět více, získat nové zkušenosti, znalosti a co možná nejlépe se umístit v soutěži Freescale Race Challenge 2012.

2 Teoretický rozbor

V této části se budu věnovat především neznámé trati, procesoru a jeho programování. Samotnou kapitolu věnuji A/D převodníkům a H-můstku pro řízení motoru. Dále se zaměřím na akcelerometr a filtrování jeho signálů.

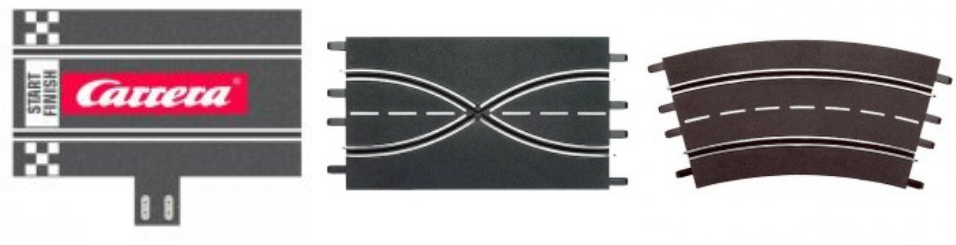
2.1 Neznámá trať

Neznámá trať obsahuje dva druhy rovinek, tři druhy zatáček, dvakrát překřížení a jeden start. Dráha neobsahuje žádné vertikální díly.

Start je tvořen optickou závorou a znázorněním bílou čárou na trati (viz první díl na obrázku 1).

Každý z dílů obsahuje dvě vodicí dráhy. Vnitřní a vnější, které mají různé vlivy na zrychlení auta v zatáčkách.

Křížení obsahuje mezeru, na které autu vypadne po určitý čas napětí.



Obrázek 1: Ukázka dílů trati [6].

2.2 Procesor MCF51JM128

Tento typ procesoru je schopný pracovat při rychlosti 50,33 MHz. Jako taktovací signál budu využívat 12 MHz krystal. Operační paměť RAM má velikost 16kbitů. Mikrokontrolér obsahuje řadu prvků, které jsou pro přehlednost uvedeny v tabulce 3. Procesor musí být napájen 3,9 až 5,5 V, aby vnitřní stabilizátor a logika fungovala na 3,3 V. Dále obsahuje sedm registrů většinou po osmi bitech, které jsou přivedeny na jednotlivé piny procesoru. Tyto takzvané porty mohou být nastaveny jako vstupní nebo výstupní. Některé z nich ovšem mohou mít i jiné funkce, jako je například vnější přerušení, výstup timeru a další. Přikládám obrázek 18 vnitřní struktury mikroprocesoru, z které je patrné rozmístění a funkce jednotlivých pinů [1].

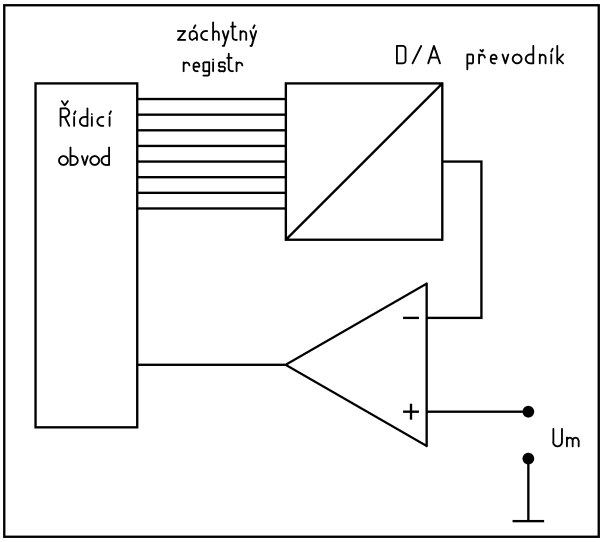
Časovač nebo-li timer – TMP, umožňuje pulsně šířkovou modulaci (dále jen PWM) s různým zarovnáním signálů nebo obyčejné časově závislé funkce. Tento typ mikrořadiče vlastní osmikanálový timer, který obsahuje 16bitový čítač. Každý kanál může být nastaven na různé funkce. Časový signál je využit z procesoru a předděličky nebo z externího signálu. Timer může při přetečení způsobit přerušení.

Mikroprocesor vlastní výstupy MISO a MOSI, které nám umožňují přenášet data na různá datová média nebo pomocí nich můžeme mikrořadič programovat. Dále obsahuje vstupy USB-DN a USB-DP pro komunikaci po USB. Zároveň nám umožňuje programování přímo z PC [1].

2.3 A/D převodníky

V mé práci jsou převodníky integrovány v procesoru (existují i jako speciální integrované obvody). A/D převodníky se používají pro převod z analogové hodnoty signálu na digitální hodnotu. V tomto typu mikrokontroléru je použit převodník s postupnou aproximací.

Princip tohoto převodníku spočívá ve využívání metody půlení intervalu a následně pak zjišťování, ve které polovině se měřené napětí nachází. Tuto polovinu potom znovu rozdělíme a znovu provedeme zjištění. Po každém dělení se určí hodnota dalšího bitu. Tento postup opakujeme tak dlouho, než je zjištěno napětí, které nejvíce odpovídá měřenému. Zde je blokové schéma tohoto typu převodníku 2 [7].



Obrázek 2: Blokové schéma převodníku [7].

Součástí je číslicově analogový převodník, který zpětně převádí výstupní slovo na analogové srovnávací napětí. Toto napětí, které se mění v každém kroku, je porovnáváno v napěťovém komparátoru s vstupním analogovým napětím. Vlastnosti tohoto typu převodníku [7]:

- A/D převodníky s postupnou aproximací mohou mít značnou rychlost převodu a velmi dobrou rozlišovací schopnost,
- doba měření dána rychlostí řídicí jednotky, D/A převodníku a komparátoru, je však nižší, než u paralelního převodníku.

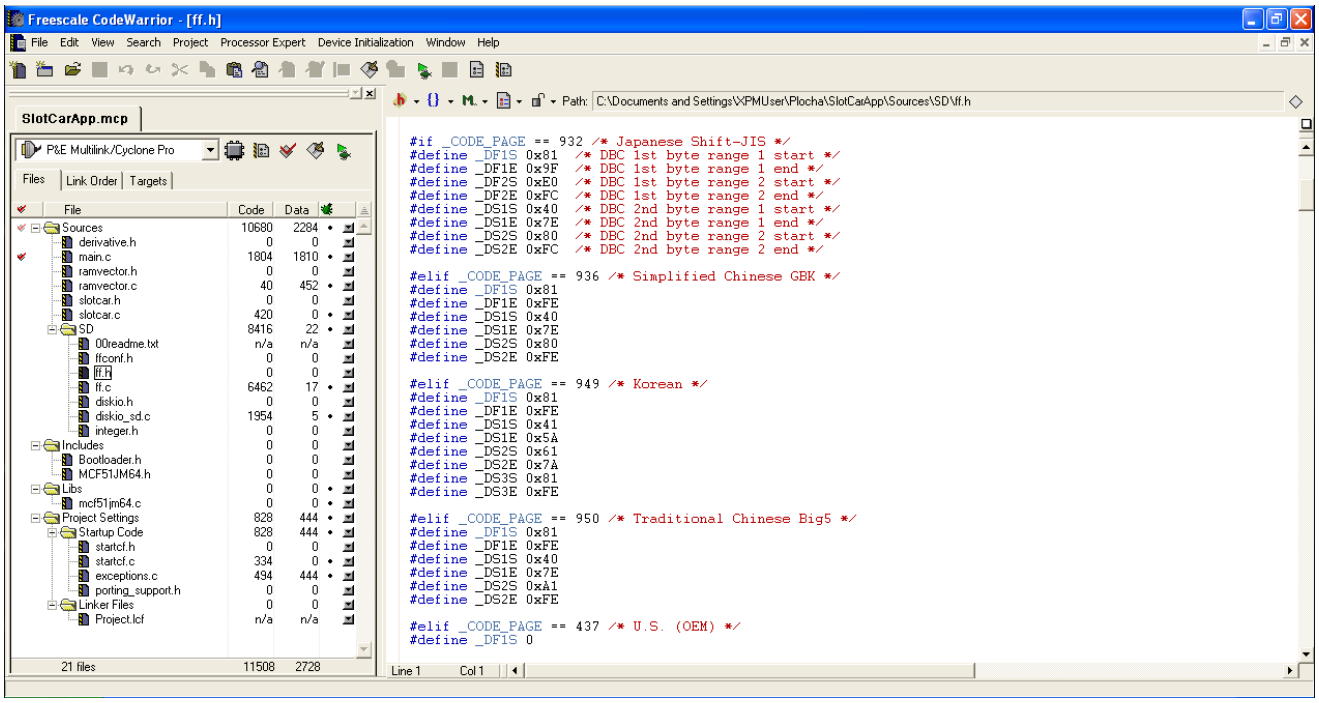
Vlastnosti převodníku procesoru MCF51JM128 [1]:

- převodník s postupnou aproximací s rozlišovací schopností 12 bitů,
- až 28 analogových vstupů,
- jednoduchý (návrat k nečinnosti po jednom převodu) nebo opakovaný převod,
- konfigurovatelná vzorkovací frekvence,
- obsahuje příznak dokončení převodu a přerušení,
- hodinový vstup volitelný až ze čtyř zdrojů,
- volitelné asynchronní spouštění,
- automatické porovnávání s nastavenou hodnotou v programu,
- teplotní čidlo.

2.4 CodeWarrior 6.3

CodeWarrior Development Studio (znázorněné na obrázku 3) je komplexní integrované vývojové prostředí (IDE), které poskytuje vizuální a automatizované programovací prostředí pro procesory v jazyce C. CodeWarrior Development Studio obsahuje následující sadu nástrojů [9]:

- Integrované vývojové prostředí (IDE)
- Projekt Wizard, Project Manager
- Processor Expert
- Zařízení pro inicializaci
- Editor, překladač, linker a makra
- Grafické ladění pomocí debugeru
- Simulátor s vizualizaci dat a vstupů/výstupů



Obrázek 3: Ukázka CodeWarrior 6.3.

- Flash programovací nástroje

Přenos samotného programu do procesoru po USB zajišťuje **bootloader**. Bootloader je softwarová aplikace předprogramovaná v procesoru. Po připojení k PC se mikrokontrolér chová jako flash paměť. Přeložený program se uloží do paměti a poté se USB odpojí. Při přiložení napájecího napětí se začne program vykonávat.

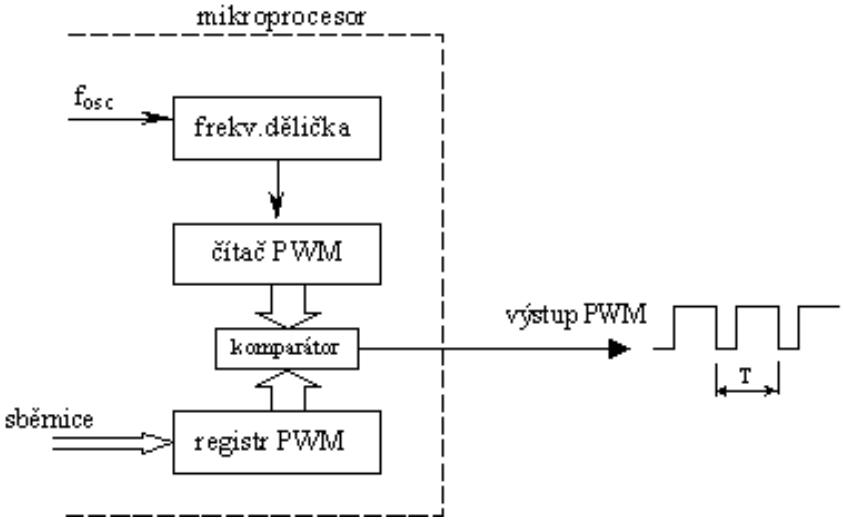
2.5 Řízení SS motoru pomocí PWM a H-můstku

U stejnosměrného motoru lze řídit rychlost pomocí napětí. Směr otáčení lze změnit přepólováním napětí na rotoru. Nejlepším způsobem, jak tento motor řídit je použití pulsně šířkové modulace v kombinaci s H-můstkem.

Pulsně šířková modulace, neboli PWM (Pulse Width Modulation), je modulace pro přenos analogového signálu pomocí dvouhodnotového signálu. Jako dvouhodnotová veličina může být použito například napětí, proud, nebo světelný tok. Signál je přenášen pomocí střídy (poměr časů kdy je v signál v log. 1 a v log. 0). Pro demodulaci takového signálu pak stačí dolnofrekvenční propust. Vzhledem ke svým vlastnostem je pulsně šířková modulace

často využívána ve výkonové elektronice pro řízení velikosti napětí nebo proudu. Kombinace PWM modulátoru a dolnofrekvenční propusti bývá rovněž využívána jako levná náhrada D/A převodníku [5].

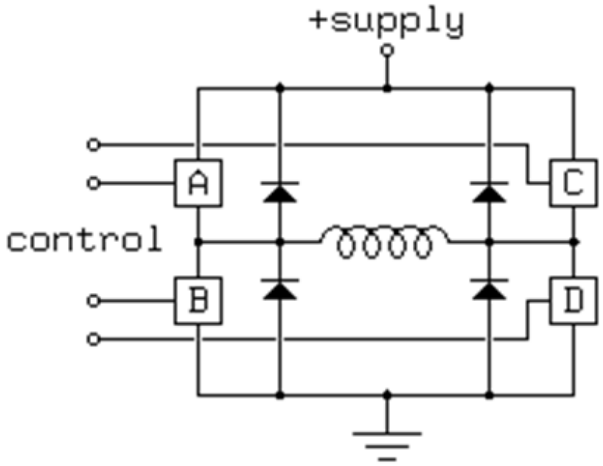
PWM může v procesoru pracovat podobně jako je znázorněno na obrázku 4.



Obrázek 4: Schéma PWM v mikroprocesoru [5].

Do frekvenční děličky se přivádí hodinový signál **fosc**, který se může ponechat nezměněn nebo se vydělí dělicí konstantou. Dělicí konstantu lze obvykle zvolit nastavením příslušného řídicího registru. Místo vnitřního hod. signálu lze použít i externí zdroj hod. signálu. Čítač PWM je volně běžicí čítač, který čítá hodinový signál z frekvenční děličky. Obsah tohoto čítače je komparátorem porovnáván s obsahem registru PWM, do kterého programově nastavíme požadovanou hodnotu. Na výstupu komparátoru dostáváme PWM signál – jsou-li obsahy čítače a registru stejné, je na výstupu log. 0, jsou-li různé, je na výstupu log. 1. Střída výstupního signálu je tedy úměrná hodnotě zapsané v registru PWM. Perioda výstupního signálu je pevná a je dána čítaným signálem a módem čítače PWM [5].

Pro přepólování motoru a pro spouštění větších proudů do motoru pomocí menších proudů z procesoru se používá můstkové zapojení většinou unipolárních tranzistorů (jako je znázorněno na obrázku 5). Díky takovému zapojení můžeme motor řídit do obou směrů a zároveň ho můžeme brzdit.



Obrázek 5: Zapojení H-můstku [4].

A, B, C, D znázorňují jednotlivé spínací prvky a cívka znázorňuje rotor motoru. Jednotlivé možnosti sepnutí jsou uvedeny pro přehlednost do tabulky 1.

Tabulka 1: Stavový popis H-můstku [3].

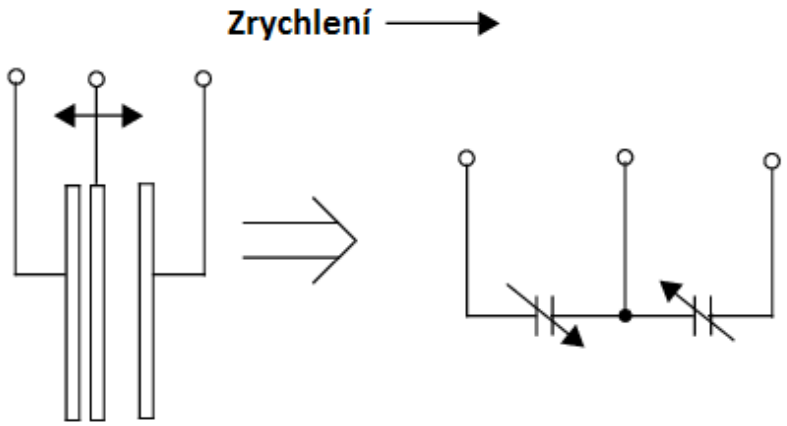
A	**B**	**C**	**D**	**Motor**
0	0	0	0	stojí
1	0	0	1	vpřed
0	1	1	0	vzad
0	1	0	1	brzda
1	0	1	0	brzda

2.6 Akcelerometr a Kalmanova filtrace

Akcelerometr je přístroj, který měří vibrace nebo zrychlení při pohybu struktur (konstrukcí, části strojů a pod.). Síla způsobující vibrace nebo změnu pohybu (akceleraci) působí na hmotu snímače.

Struktura a funkce akcelerometru je založena na proměnné kapacitě tříelektrodového vzduchového kondenzátoru. Využívá se zde známé nelineární závislosti kapacity C na vzdá-

lenosti elektrod kondenzátoru. Pokud tedy jednu elektrodu necháme pohyblivou a její pohyb bude závislý na působícím zrychlení, získáme kapacitní akcelerometr. Takový princip je znázorněn na obrázku 6 [2].



Obrázek 6: Akcelerometr [2].

Protože je signál ze samotného akcelerometru značně zašuměný je třeba ho filtrovat. Pro toto použití se nejvíce hodí **Kalmanova filtrace**.

Kalmanova filtrace je speciální matematický aparát (algoritmus) pro filtraci signálů v časové oblasti. Výhodou tohoto systému je schopnost získat čistý signál a hodnoty ze zašuměného signálu nebo jinak znehodnoceného souboru hodnot, i bez jakéhokoliv poznatku o rušení. Prakticky lze takto matematicky zjistit hodnoty, které jsou přímým měřením těžko zjistitelné, protože se při samotném aktu měření do získaných hodnot indukují chyby měřicích přístrojů nebo okolní působící šum a rušení. Jelikož není třeba počítat Fourierovu transformaci je proces filtrace rychlý [8].

Když to vezmeme od začátku, lze celý Kalmanův algoritmus popsat jako průměrování odchylek naměřených hodnot od odhadovaných hodnot a jejich nejistot, tedy pravděpodobností, že naměřená hodnota je správná. Například si lze představit jachtu plující po moři a my se snažíme určit její přesnou polohu a směr plavby. V určitých měřicích okamžicích zjišťujeme její okamžité polohy s nějakou chybou (přesností). Kalmanův algoritmus je pak schopen hodnoty zpřesnit tak, že určuje rozdíly odhadnuté (predikované) polohy a provádí průměrování těchto chybových hodnot se zapomínáním starších vzorků. Na základě průběhu předchozích chyb, resp. jejich rozptylu, dochází k odhadu nejbližších následujících poloh [8].

3 Realizace

Tato kapitola bude věnována konstrukci auta, řídicímu algoritmu a naměřeným hodnotám. Auto na obrázku 15 obsahuje desku plošného spoje s procesorem (viz obrázek 22) a plošný spoj se senzory a záložním zdrojem (viz obrázek 23).

3.1 Hardware

Celé auto je řízené procesorem MCF51JM128RM, který přijímá signály ze senzorů a pomocí H-můstku MC33931 ovládá otáčky motoru. Procesor lze programovat přes USB, zároveň komunikuje prostřednictvím sériového periferního rozhraní (SPI – Serial Peripheral Interface) s paměťovým médiem v našem případě SD kartou.

3.1.1 Zapojení procesoru

Tabulka 2: Zapojení procesoru.

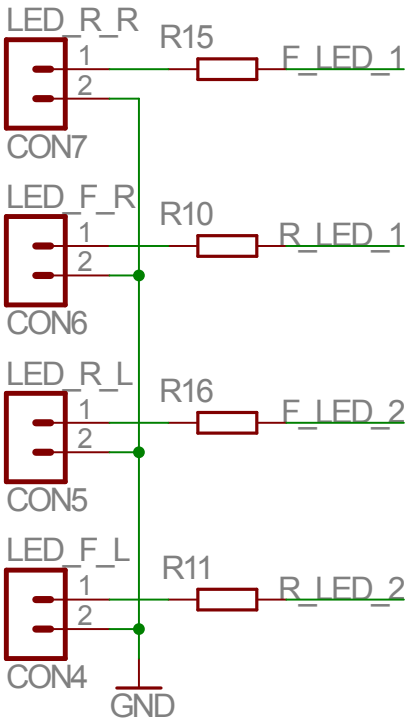
Název pinu	**Zapojení**
PTE – MOSI MISO CLK SS	Vývody pro zapojení SD karty.
R-LED F-LED	Přední a zadní světla připojené přes odpory.
IN1,2 D1,2 SF FB	Řízení H-můstku (piny IN1,2 jsou zapojeny na PWM)
XTAL EXTAL	Připojení 12 [MHz] krystalu.
RESET	Resetování procesoru při připojení napájení realizované RC článkem.
SW	Přepínač.
PTB – MOSI MISO CLK SS	Umožňuje programování procesoru pomocí programátoru (např. přeprogramování bootloaderu).
INT1,2	Vstup z napájení přes odporové děliče na A/D převodník, umožňuje měřit napájení tratě nebo USB.
X Y Z	Měření ve třech osách z akcelerometru.
USB_DN USB_DP	Datové piny pro USB programování.

Procesor je napájen 3,3 Volty ze stabilizátoru. Jednotlivé piny mají různé funkce, využití těchto funkcí je znázorněno pomocí obrázku 19 a tabulky 2.

Akcelerometr je připojen přes vyhlazovací kondenzátory přímo na A/D převodník. INT1 a INT2 měří pomocí A/D převodníku napájení a rozhoduje, která z části programů se má vykonávat (program načítání trati nebo bootloader).

3.1.2 Zapojení předních a zadních světel

Světla jsou zapojena podle schematu na obrázku 7. LED_R_R a LED_F_R jsou zadní a přední pravá světla a LED_R_L a LED_F_L jsou levá zadní a přední světla. Jako přední světla jsem použil supersvítivé nízkopříkonové diody bílé barvy a jako zadní jsem použil 2 mA červené diody. Proto pro přední diody byl zvolen menší odpor (z důvodu většího odběru) a pro zadní světla odpor menší.

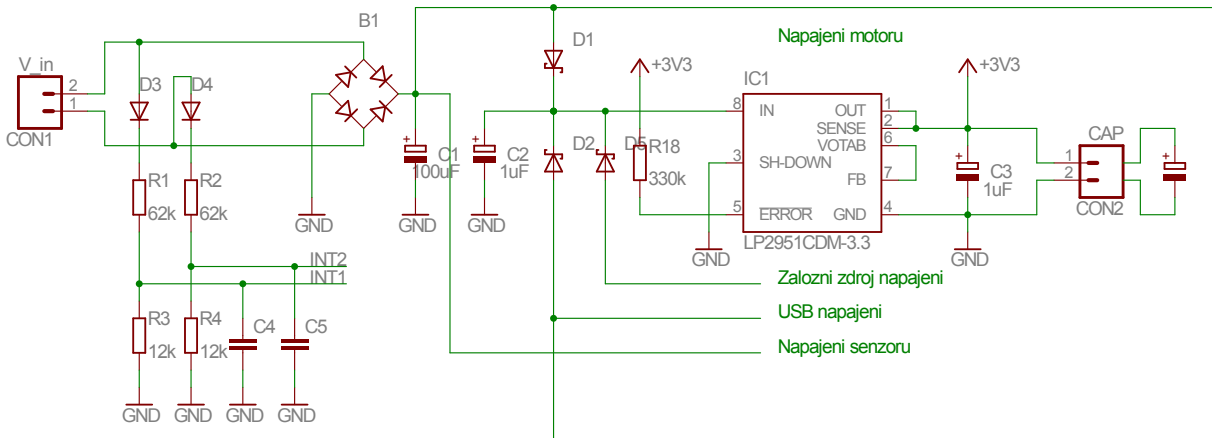


Obrázek 7: Zapojení světel.

LED diody byly používány pro signalizaci různých částí programu. Například pro zobrazení křížení, pro zobrazení vjezdu a výjezdu do rovinky nebo při zrychlování a zpomalování.

Diody jsou přidělány na speciálních plošných spojích, které jsou tvořeny přesně do držáků v podvozku auta (obr. 16, 17).

3.1.3 Stabilizace napájení



Obrázek 8: Stabilizace napájení.

Zapojení obsahuje dva stabilizátory LP2950. Jeden je speciálně vyhrazený pro senzory z důvodu většího odběru.

CON1 je připojený na kartáče, které dosedají na trať a napájí auto. Z konektoru vede napětí na dělič pro snímání a na můstkový usměrňovač. Usměrňovač zajistí ochranu při přepólování (nasazení auta do protisměru).

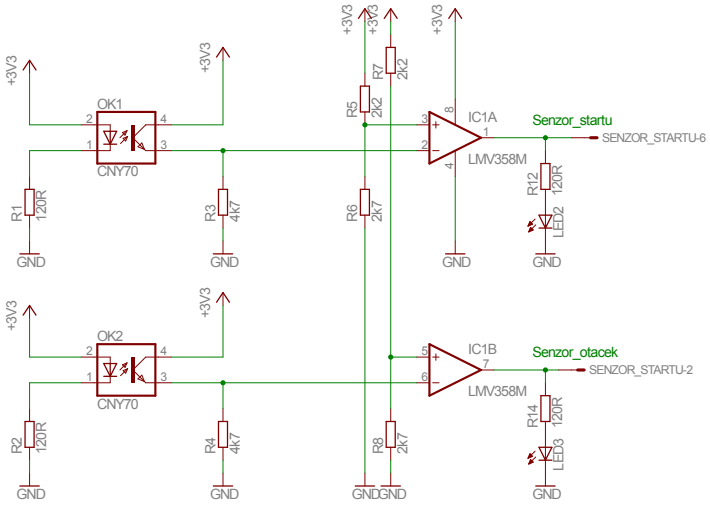
Pokud je auto na dráze, proud vede přes zpětnou diodu na napájení motoru a senzorů. Je-li auto napájeno z USB, proud jde opět zpětnou diodou, totéž platí i u záložního zdroje. Zpětné diody zajišťují aby proud netekl směrem do zdroje pokud je auto napájeno z více zařízení najednou.

Musíme dát pozor, protože procesor nerozezná rozdíl mezi napětím ze záložního zdroje a USB. Je nutné přidat rozpojovací kontakt a při každém programování vypínat napájení ze záložních baterií. Jinak by baterie zůstaly sepnuté a brzy by se vybily.

Ke konektoru **CON2** je připojen kondenzátor o kapacitě $C = 1$ mF, který zajistí potřebný čas pro zapnutí záložního zdroje při běhu procesoru.

3.1.4 Senzory

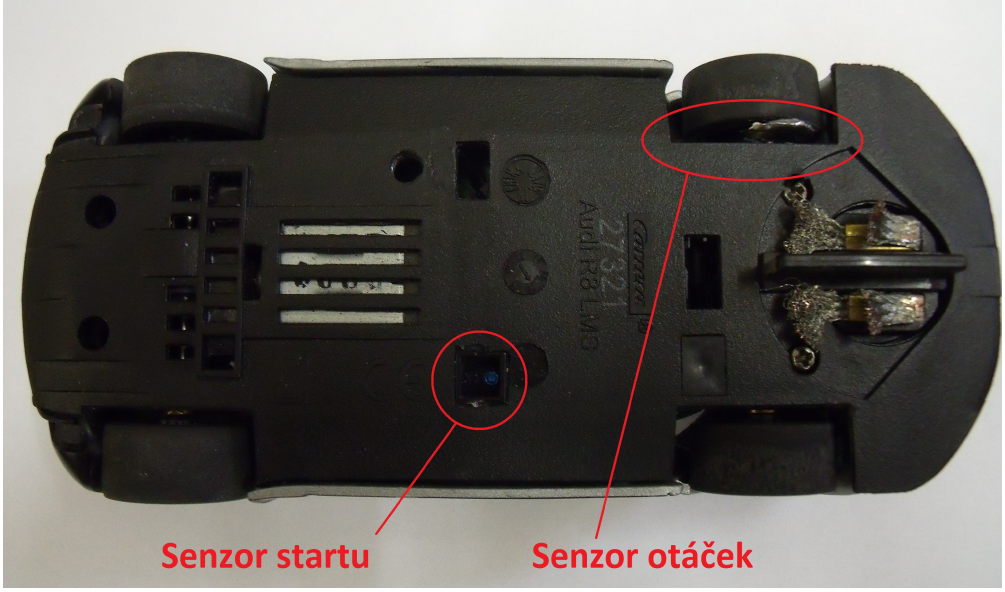
Senzory pracují na principu odrazu světla. Jelikož je start na trati vyznačen bílou barvou je možné ho měřit. Pro měření otáček je také použito optické čidlo. Toto čidlo je připevněno tak, aby směřovalo na vnitřní stranu předního kola. Na části kola je potom přilepen alobal, který zajistí odraz. Když je kolo natočeno v části, kde alobal není, čidlo ukáže jinou hodnotu. Umístění senzorů je znázorněno na obrázku 10.



Obrázek 9: Zapojení senzorů.

Za senzory startu a otáček byly zvoleny optické závory **CNY70**. Byly vybrány díky své velikosti, dostupnosti a orientaci infračervené diody (dále jen IR) a fototranzistoru v jedné rovině. Fototranzistory jsou připojeny ke Schmittovu klopnému obvodu, jak je patrné z obrázku 9. Při změně světelného toku se změní hodnota napětí na výstupu. To je dále porovnáváno s experimentálně nastaveným napětím na kladném vývodu operačních zesilovačů. Toto napětí je různé pro oba senzory a je nastaveno pomocí odporového děliče. Pro senzor startu je to dělič R5, R6 a pro senzor otáček R7, R8. Na výstupu poté dostaneme vyhlazený obdélníkový signál, který indikují LED diody. Při přejetí startu je senzor v log. 0 a v neaktivním stavu v log. 1. Tento signál je dále připojen k procesoru na porty PTD6 a PTD7 (viz obr.19).

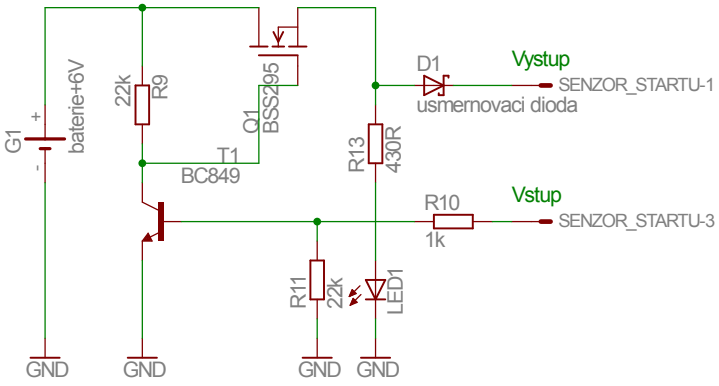
Z důvodu velkého odběru senzorů (až 80 mA) byl použit druhý stabilizátor. Jak je vidět na obrázku 8. Senzory pracují pouze při napájení z dráhy a nikoli při napájení ze záložního zdroje. Tento problém se vyskytne zejména na křížení, proto je nutné s tím počítat při návrhu samotného programu.



Obrázek 10: Umístění senzorů.

3.1.5 Záložní napájení

Aby se procesor při výpadku auta z dráhy nebo při přejetí překřížení nerestartoval a neztratil tak naměřená data, rozhodl jsem se zabudovat záložní zdroj. Záložní zdroj tvoří dvě baterie typu **2032**, každá o napětí 3 V. Dále jsou to dva tranzistory, jeden unipolární p kanál a druhý NPN bipolár (obr. 11). Po připojení log. 1 na vstup se otevře unipolární tranzistor a záložní zdroj je zapnut (zapnutí signalizuje LED dioda).



Obrázek 11: Záložní zdroj.

Baterie mají kapacitu 200 mAh. To dostatečně vystačí na přejezdy překřížení a na určitou dobu mimo trať. Je nutné vypínat doplňková zařízení a zbytečně neplýtvat energií (například vypínání předních a zadních světel atd.).

3.2 Software

Tato kapitola pojednává o funkci algoritmu. Jsou zde uvedeny důležité části ukládání, načítání a synchronizace trati. Dále se budu zabývat správným vyčítáním ze senzorů a ukládáním na SD kartu.

3.2.1 SD karta

SD karta (Secure Digital) je flash paměťové médium. V mém případě používám microSD kartu o velikosti 2 GB. Pro komunikaci s procesorem byla zvolena sběrnice SPI.

Pro programování jsem využil knihoven **FatFs Generic FAT File System Module**. Tato knihovna je programována v jazyce **ANSI C**. Knihovna obsahuje například tyto funkce:

- `f_mount` – alokování SD karty a registrace úložného prostoru
- `f_open` – vytvoření souboru nebo otevření starého
- `f_printf` – zapsání dat do souboru (typu char)
- `f_read` – čtení ze souboru
- `f_sync` – uloží data ale do souboru se může stále zapisovat
- `f_close` – zavření souboru

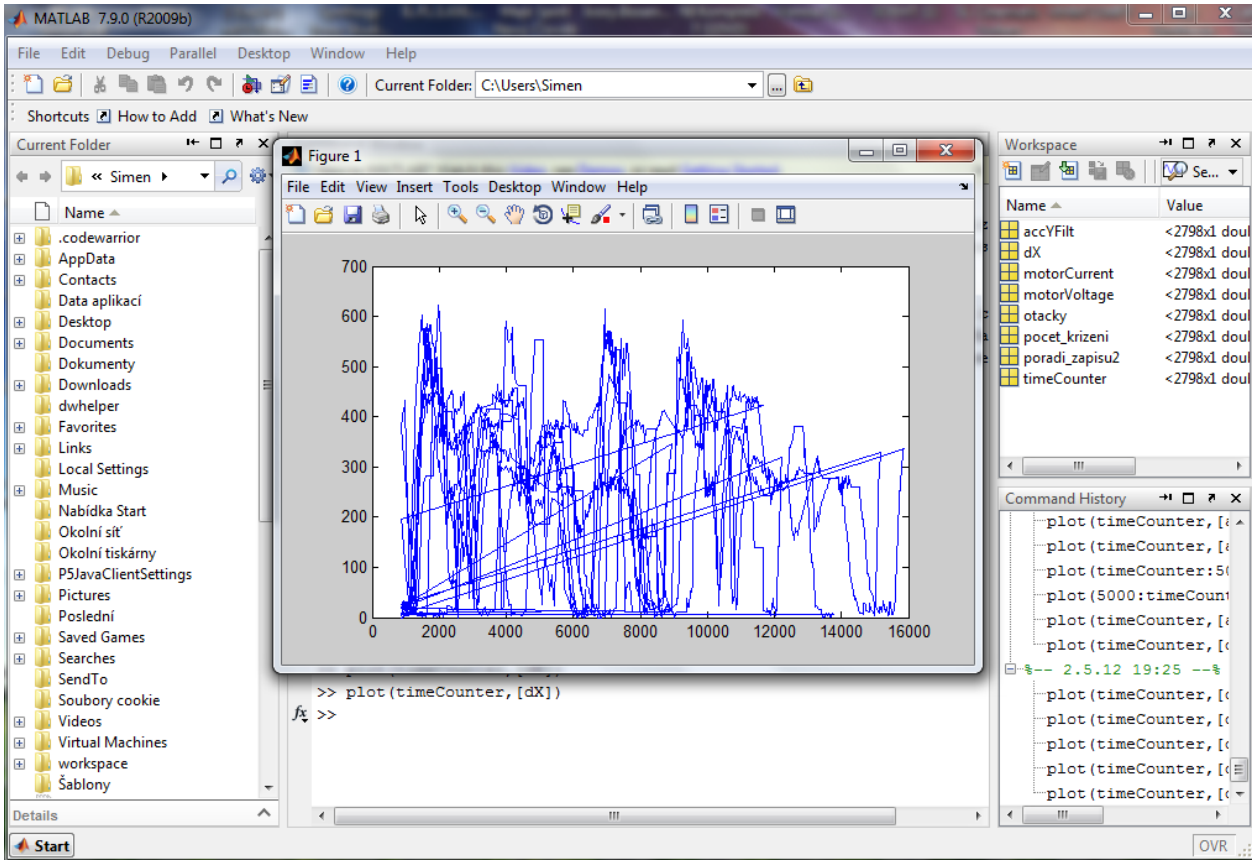
Paměť nepoužívám pro ukládání tratě z důvodu dlouhé doby čtení hodnot (kolem 100 ms). Používám ji však pro analyzování dat pomocí PC. Do karty zapíši nejdříve první řádek. Ve zjednodušené podobě programu to vypadá následovně:

```
f_mount(0, &fileSystem);
fileName[] = nazev_souboru;
f_open(&file, fileName, FA_CREATE_NEW | FA_WRITE);
f_printf(&file, %s, "timeCounter;rychlost;motorCurrent;dX;
        otacky;poradi_zapisu2;motorVoltage;pocet_krizeni");
```

Nejprve alokuji SD kartu. Nastavím jméno souboru a založím ho. Napíši první řádek (hlavičku) do souboru. Při cyklech zapisování už pouze volám funkci `f_printf` a zapisuji uložený buffer na paměťové médium. Jméno právě otevřeného souboru lze napsat jako `&file`.

Uložená data mají podobu tabulky. Pro čtení a vizualizaci naměřených dat používám prostředí **Matlab**. Data musím nejdříve přeuložit do textového souboru a poté importovat tak, že k jednotlivým vektorům přiřadím jména z hlavičky.

To mi dává řadu možností jak naměřená data zobrazovat. Pro tvorbu grafu používám příkaz `plot(timeCounter,[dX accXfilt])`. Jako časová osa slouží půlmilisekundový čítač a na osu x zobrazuji veškerá naměřená data od rychlosti až po zrychlení a počet překřížení.



Obrázek 12: Ukázka prostředí Matlab.

Na obrázku 12 je vidět základní prostředí aplikace **Matlab**. Uprostřed je vytvořený graf, na kterém je zobrazené zašumění zrychlení bez použití filtru, to ale není podstatné. Pod grafem je vidět příkazová řádka s příkazem pro vytvoření grafu. Po pravé straně vidíme názvy vektorů (matice velikosti 1 × počet prvků). V pravém dolním rohu je historie použitých příkazů.

V prostředí grafu lze pomocí nástrojů kreslit a provádět jiné úpravy. Pro vytvoření obrázku se použije menu **File** – **Save As**.

3.2.2 Základ programu

Program se skládá z několika souborů, které jsou importovány do základního souboru **main.c**. Jedná se především o bootloader, knihovny pro ovládání SD karty, základní nastavení procesoru (zejména A/D převodníků a portů), atd. Struktura základního programu vypadá ve zjednodušené podobě asi takto:

- přiřazení používaných souborů,
- alokace proměnných,
- přiřazení konstant,
- vytvoření bufferů pro ukládání na SD kartu a pro mapu,
- funkce a podprogramy,
- obsluha přerušení,
- základní načtení programu a vytvoření souboru,
- nekonečný cyklus,
- konec programu, do kterého by se algoritmus nikdy neměl dostat.

Nekonečný cyklus je vytvořen pomocí kódu jako `while(1){}`. Poté co se do tohoto cyklu program dostane, začne kontrolovat podle podmínek chod motoru. Z této smyčky se dostane pouze v případě, že dojde k přerušení nebo zavolání podprogramu.

Program obsahuje řadu přerušení jako je například:

- přerušení od timeru každých 10 ms,
- přerušení od timeru každých 0,5 ms,
- přerušení od A/D převodníku (po dokončení převodu),
- externí přerušení.

Každých 10 ms zajišťuji přepsání hodnot z proměnných do bufferu pro ukládání na SD kartu a zároveň nastavuji přepočet rychlosti (počet otáček za 10 ms).

V časových intervalech 0,5 ms nastavuji PWM podle globální proměnné (napětí motoru odpovídá velikosti konstanty). Dále každých 0,5 ms kontroluji stav senzoru otáček a stav napětí. Při výpadku napětí se inkrementuje proměnná `doba_vypadku`. V tomto přerušení se také přičítá do vytvořených časovačů.

3.2.3 Podprogram záložního zdroje

Podprogram záložního zdroje je základním stavebním kamenem celého algoritmu. Stará se nám o spouštění záložního zdroje na překřížení nebo při výpadku, dále se nám stará o měření překřížení. Při výpadku z trati zajišťuje nastavení času po který je auto aktivní a lze ho vrátit na trať. Pokud auto vypadne a nevrátím ho, auto se vypne. Zdrojový kód pro kontrolu napájení vypadá takto:

```
unsigned short stav_napajeni(void){
   if (doba_vypadku == 0)
      {return 3;}
   if (doba_vypadku <= 9 && doba_vypadku > 0)
      {zdroj_zap;
      if (GET_INT1 == 1 || GET_INT2 == 1)
         {doba_vypadku = 0;
         return 3;}
      return 7;}
   if (doba_vypadku < 200 && doba_vypadku > 9)
      {zdroj_zap;
      start = 0;
      if (GET_INT1 == 1 || GET_INT2 == 1)
         {return 1;}
      return 6;}
   if (doba_vypadku >= 200 && doba_vypadku <= 2000)
      {zdroj_zap;
      if (GET_INT1 == 1 || GET_INT2 == 1)
```

```
   {doba_vypadku = 0;
   return 0;}
return 5;}
if (doba_vypadku > 20000)
{zdroj_vyp;
return 2;}
}
```

Každý cyklus programu začíná právě kontrolou napětí pomocí tohoto podprogramu. Pokud je auto na dráze `doba_vypadku` se rovná nule. Program vrátí hodnotu 3 a program je připraven jak zapisovat, tak načítat trať.

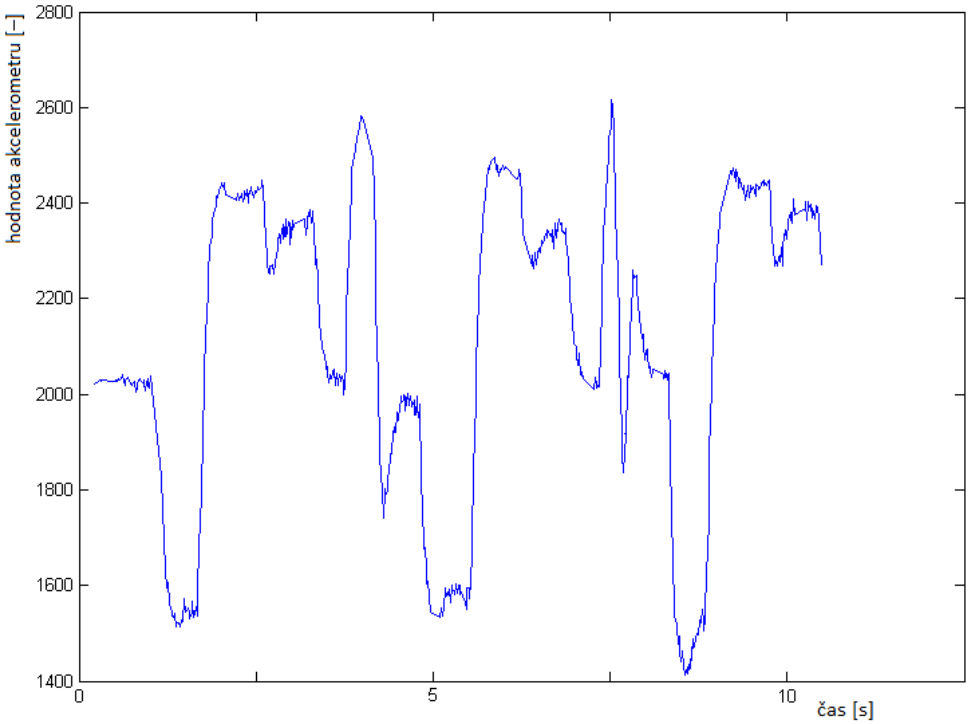
Při výpadku nebo překřížení je inkrementována proměnná `doba_vypadku` každou 0,5 ms. Pokud program vrátí hodnotu 5 nebo 6 jedná se o výpadek napětí, ale v případě hodnoty 6 není jisté, zda se jedná o ztrátu napětí zapříčiněnou křížením nebo výpadkem. Proto jsou přidány podmínky s proměnnými `GET_INT1` a `GET_INT2`, které se nastaví na log. 1 pokud je napětí na kartáčích. Přijde-li napětí z dráhy v časovém intervalu 4,5 až 100 ms podprogram vrátí hodnotu 1 a je jasné, že šlo o křížení. Pokud se napětí objeví v intervalu 100 ms až 10 s program vrátí číslo 0, to znamená, že auto vypadlo z dráhy a bylo opět vráceno zpět. Algoritmus pozná, že se má synchronizovat s tratí.

Jestliže se napětí neobjeví na kartáčích do 10 s, program vypne záložní zdroj a tím i celé auto.

Na dráze mohou být nerovnosti a může se stát, že vypadne napětí a auto změří špatně překřížení, proto je zde i podmínka, která zamezí výkyvům napětí.

3.2.4 Ukládání dat o tvaru trati

Data z akcelerometru jsou značně zašuměná, proto je nutné signál filtrovat. Pro filtrování jsem vybral Kalmanův filtr, z důvodu vcelku rychlého výpočtu. Využíván bude pouze signál v **X ose**. Po aplikaci filtru vypadají uložená data, jak je vidět na grafu 13. Z grafu je patrné, že auto jede po rovině kolem hodnoty zrychlení přibližně 2000.

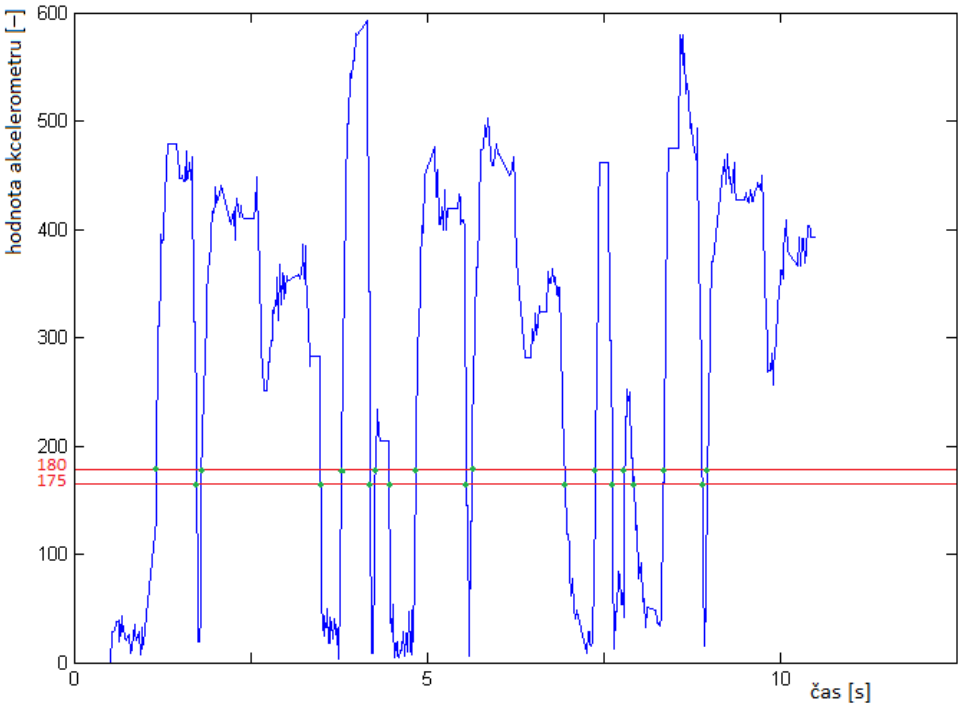


Obrázek 13: Graf zrychlení.

Pro ukládání trati jsem zvolil následující postup:

- Mapu jsem založil v procesoru, protože ukládání trati na SD kartu by bylo zdlouhavé. Mapa je tvořena maticí hodnot 3×150, aby bylo možné ukládat až 150 aktuálních hodnot zrychlení, otáček a času.
- Nastavil jsem konstantní rychlost auta.
- Vytvořil absolutní hodnoty z naměřených dat akcelerometru a odečetl stejnosměrnou složku.

- Ukládat jsem začal při průjezdu startem a to tak, že jsem ukládal do mapy první hodnotu větší než 180 a první menší hodnotu než je 175. Pro názornost uvedené v grafu 14.



Obrázek 14: Graf absolutní hodnoty zrychlení.

Tímto způsobem jsem docílil toho, že jsou uloženy hodnoty pouze začátků a konců zatáček. (Zároveň konec zatáčky znamená začátek rovinky a začátek další zatáčky znamená konec rovinky.) Mapa je tak vcelku jednoduchá a jsem schopen načítat dvoje hodnoty najednou a řídit tak velice rychle. Mohu také pomocí různých výpočtů měnit průběh rychlosti v jednotlivých úsecích dráhy.

3.2.5 Načítání trati

Při průjezdu startem se nuluje počítání časovače a otáček. Po uložení trati se načtou naráz dvoje hodnoty (začátku a konce zatáčky). Pak se řídí rychlost pro jednotlivé úseky. Při výjezdu ze zatáčky se načtou opět nové dvě hodnoty.

Když se hodnota adresy právě načtených dat rovná hodnotě počtu prvků v mapě mínus jedna, tak auto jede poslední rovinkou před startem.

Závěr

V bakalářské práci se mi podařil splnit hlavní cíl – navrhnout a vytvořit samořídicí auto na autodráhu společně s algoritmem pro mikroprocesor. Firma Freescale mi dodala hotový plošný spoj s procesorem 22. Auto nejdříve načetlo a uložilo dráhu a poté ji projelo maximální povolenou rychlostí tak, že nevypadlo z dráhy.

V průběhu realizace této práce se objevila řada problémů, které jsem musel vyřešit. Uvedu zde některé z nich.

Po připojení senzorů ke zbytku auta, začalo vypadávat napětí a motor se točil konstantní rychlostí. Problém byl způsoben velkým odběrem auta a následným přehřátím stabilizátoru, který má tepelnou pojistku. Přidal jsem druhý stabilizátor pro senzory a tímto byl nedostatek odstraněn.

Bohužel úpravou přestaly fungovat senzory při výpadku napětí. Tomu jsem zamezil úpravou algoritmu.

Senzor startu měřil spolu se startem také odraz od kolejí s napájením na dráze (od jejich lesklého povrchu). Tuto chybu jsem z části odstranil úpravou softwaru, za použití časovače. Lepším řešením by bylo použít druhý senzor, který jsem z hmotnostních důvodů nemohl použít.

Při nastavování záložního zdroje se vyskytla chyba v algoritmu. Záložní zdroj se po opětovném nasazení na dráhu nevypínal. Musel jsem analyzovat problém v programu a řešit spínání zdroje jinou cestou.

Zajímavé bylo sledovat jednotlivé průběhy zrychlení, otáček a rychlosti. Naučil jsem se pracovat s programem **CodeWarrior** pro programování procesoru, s aplikací **TeXnicCenter** pro tvorbu strukturovaného textu. Zdokonalil jsem své schopnosti v návrhovém prostředí **Eagle**. Osvojil jsem si problematiku programování mikroprocesorů a využívání knihoven pro funkce inicializace SD karty aj..

I přes veškeré problémy se mi podařilo sestavit funkční model auta. V soutěži Freescale Race Challenge 2012 jsem se umístil na čtvrtém místě ze sedmi týmů. Získané zkušenosti jistě využiji nejen při studiu, ale i v budoucím zaměstnání.

Literatura

[1] MCF51JM128 – Microcontroller Reference Manual. [cit. 2012-01-03]. Dostupné z: http://www.freescale.com/files/32bit/doc/ref_manual/MCF51JM128RM.pdf

[2] MMA7361L – ±1.5g, ±6g Three Axis Micromachined Accelerometer. [cit. 2012-01-04]. Dostupné z: http://www.freescale.com/files/sensors/doc/data_sheet/MMA7361L.pdf

[3] MC33931 – A Throttle Control H-bridge. [cit. 2012-01-04]. Dostupné z: http://www.freescale.com/files/analog/doc/data_sheet/MC33931.pdf

[4] JONES, Douglas. Stepping Motors [online]. [cit. 2012-01-04]. Dostupné z: http://www.divms.uiowa.edu/j̃ones/step/circuits.html

[5] DH servis – Pulsně šířková modulace [online]. [cit. 2012-01-05]. Dostupné z: http://www.dhservis.cz/psm.htm

[6] Carrera [online]. [cit. 2012-02-29]. Dostupné z: http://www.carrera.cz/

[7] AŘT -2. AD převodníky. [online]. [cit. 2012-01-06]. Dostupné z: www.sketa-shop.ic.cz/maturita/ART/ART2.doc

[8] Měření a regulace – Co je to Kalmanova filtrace? [online]. [cit. 2012-02-30]. Dostupné z: http://automatizace.hw.cz/clanek/2007042901

[9] CodeWarrior Development Tools [online]. [cit. 2012-01-04]. Dostupné z: http://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME

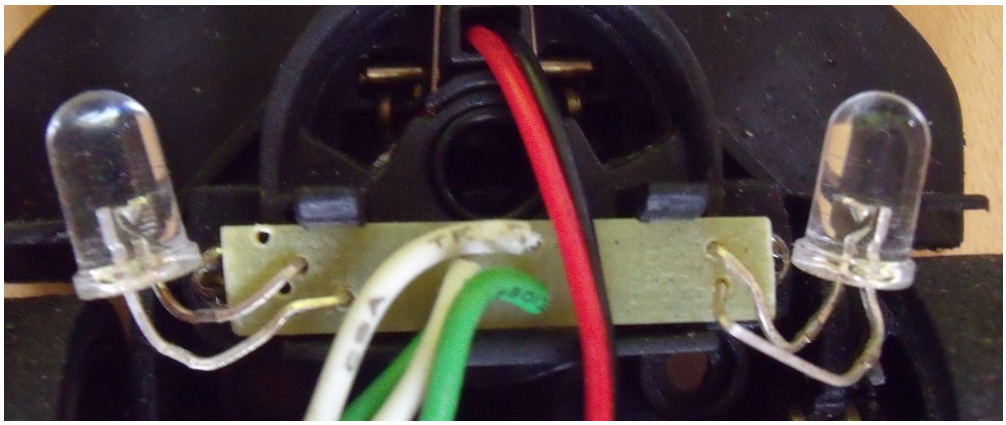
[10] Freescale Race Challenge 2012 [online]. Dostupné z: http://www.hw.cz/teorie-a-praxe/mimochodem/freescale-race-challenge-2012-soutez-samoridicich-auticek-na-autodrahu

Přílohy

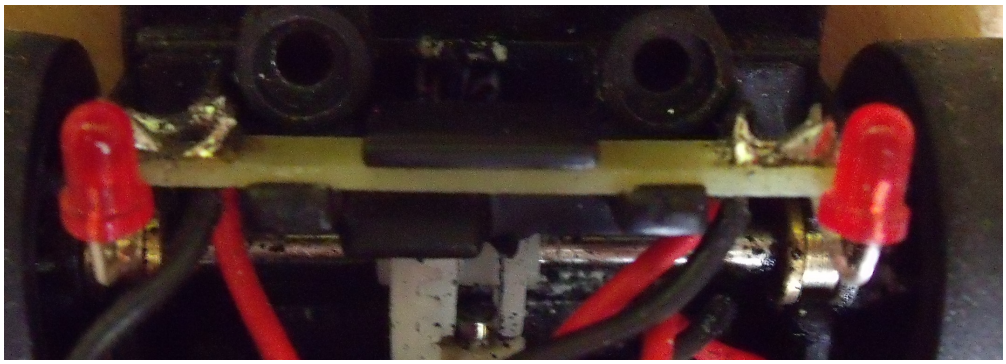
Přikládám potřebné obrázky a CD, které obsahuje následující složky:

- práci vytvořenou v prostředí LaTeX,
- schémata navržená v programu Eagle,
- algoritmus naprogramovaný v návrhovém prostředí CodeWarrior,
- datasheety základních součástek.

Obrázek 15: Auto na autodráhu.



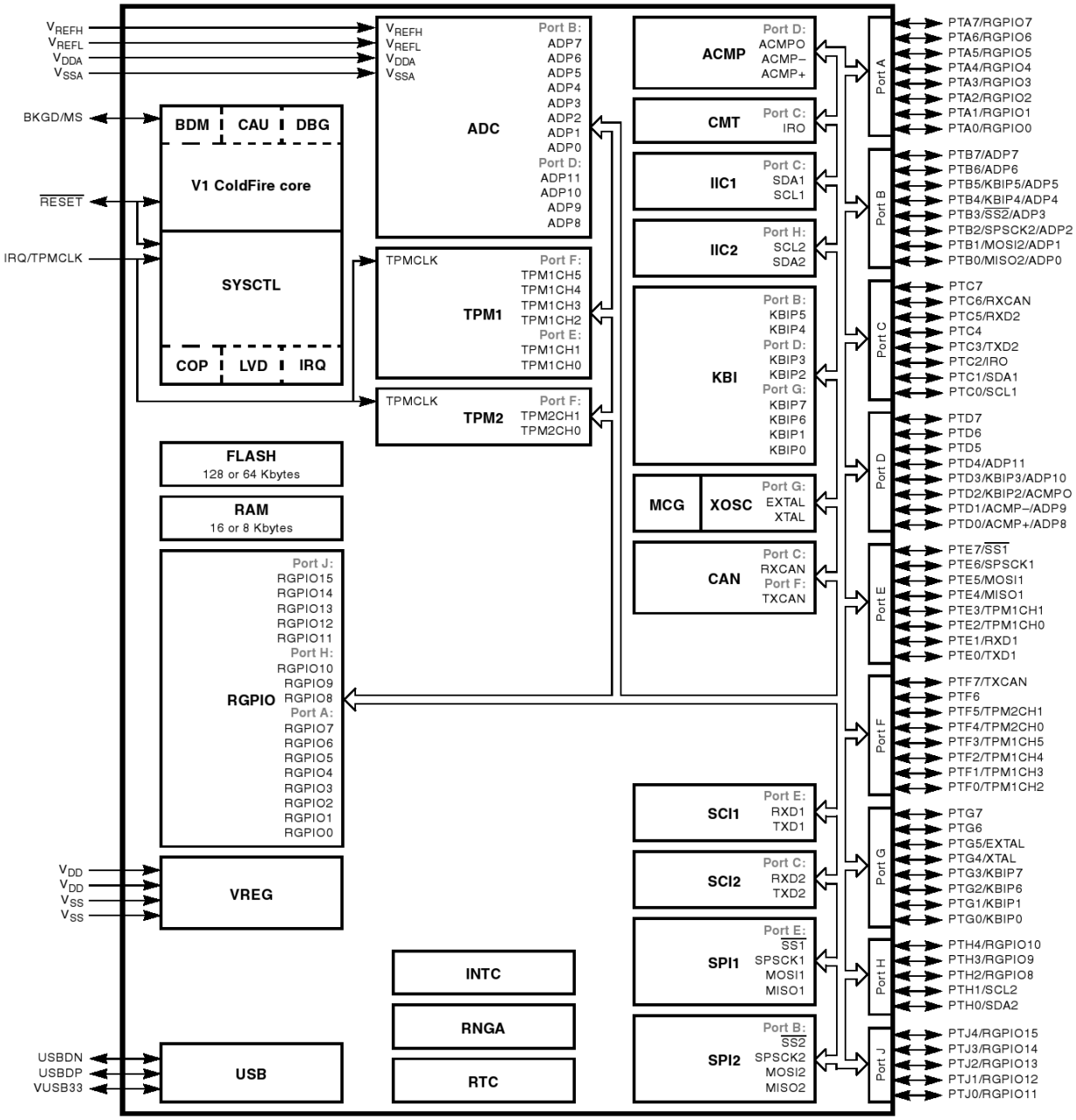
Obrázek 16: Přední světla.



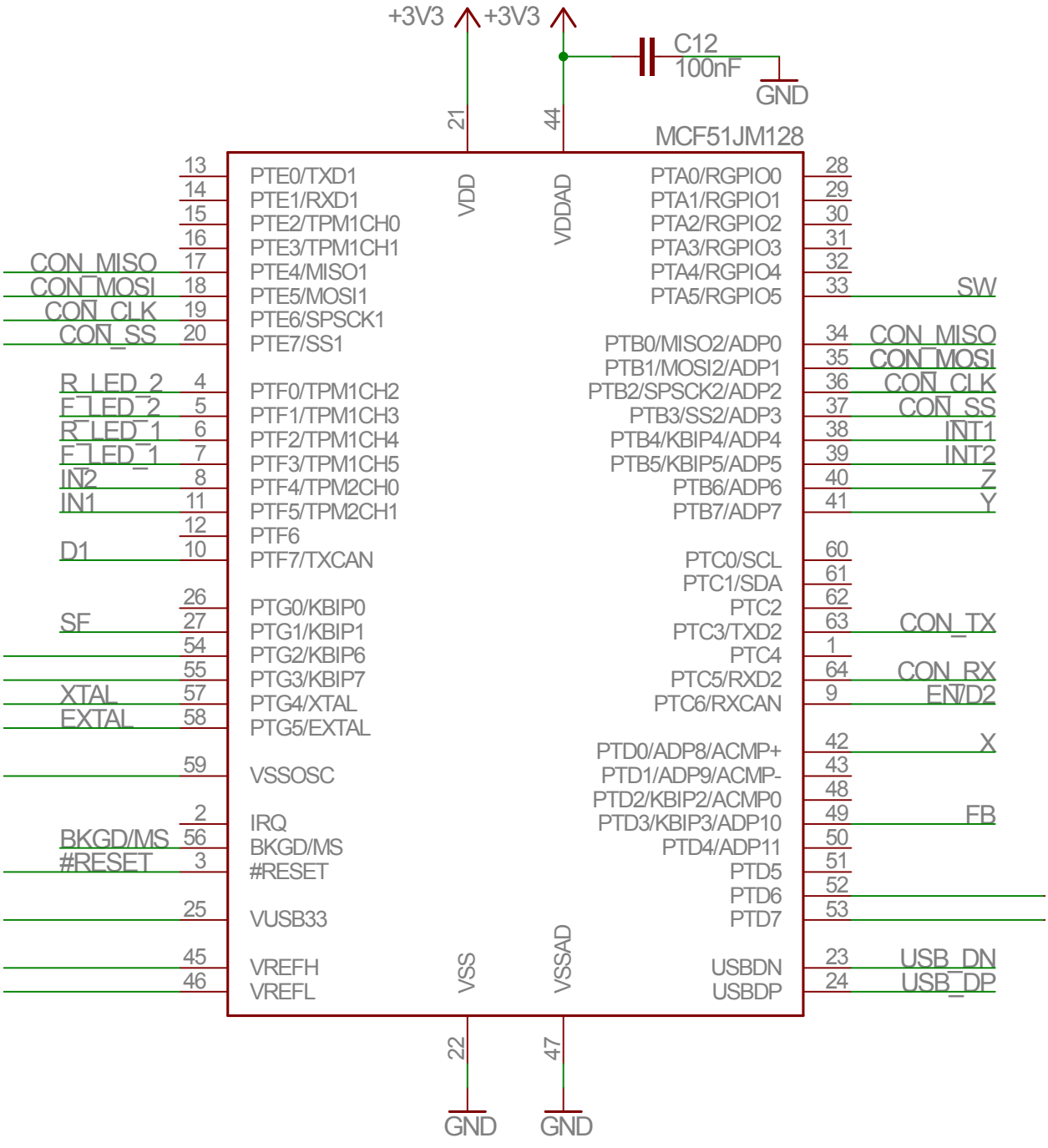
Obrázek 17: Zadní světla.

Tabulka 3: Funkce procesoru [1].

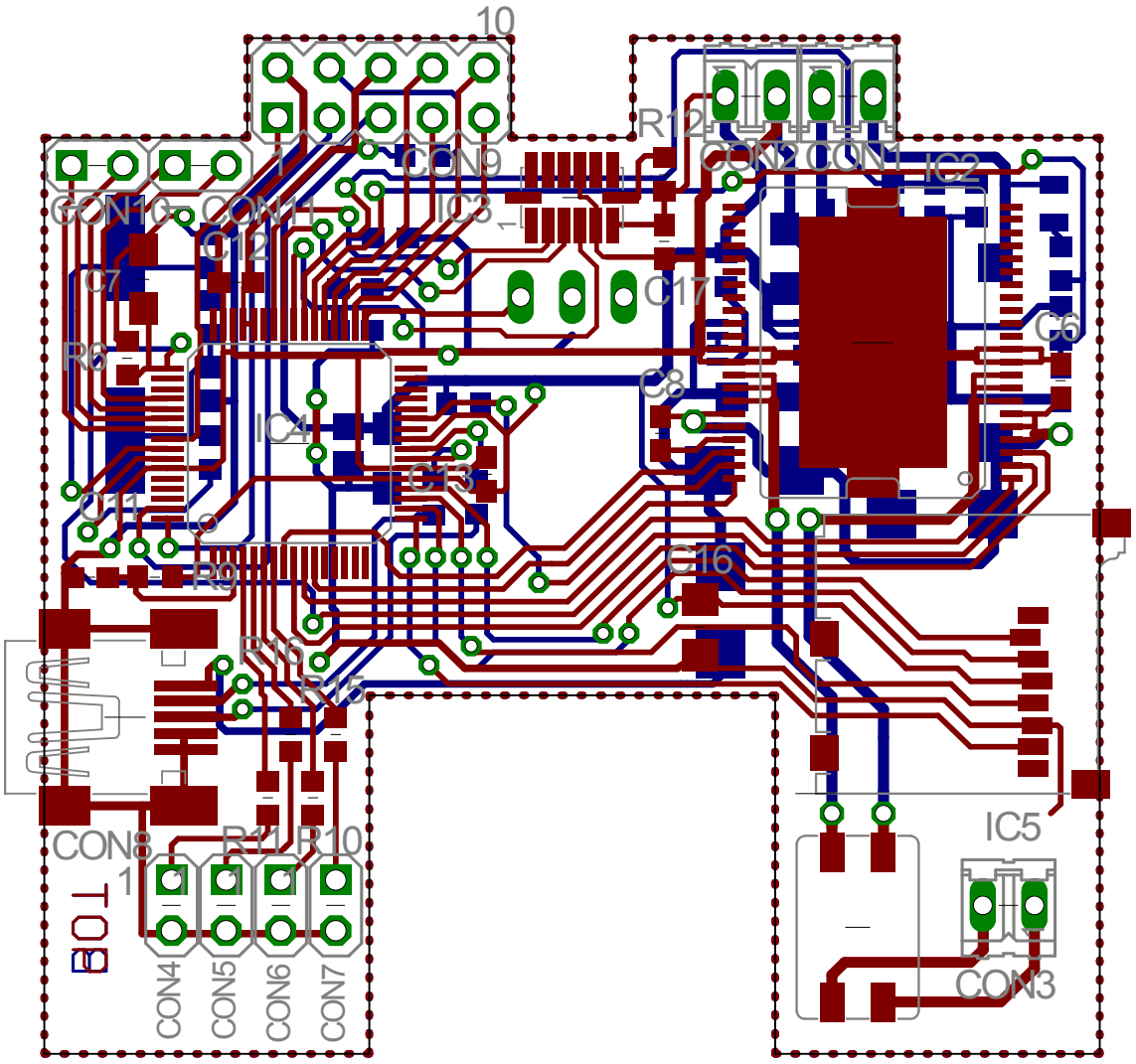
Zkratka	**Funkce**
CF1CORE	Vykonává programy a přerušení.
BDM	Obsahuje jeden pin pro ladění
DBG (debug)	Poskytuje možnost ladění a simulace
SYSCTL	Poskytuje LVD, COP, vnější zdroj přerušení,atd.
FLASH	Obsahuje paměť pro program, konstanty a proměnné
RAM	Operační paměť
RGPIO	Umožňuje přístup rychlostí hodin k I/O
VREG	Kontroluje hodnoty napětí v zařízení
USB	Možnost programování přímo přes USB
ADC	12bitový analog-digitální převodník
TPM1, TPM2	Poskytuje různé časování včetně PWM
CF1INTC	Kontroluje a udává prioritu přerušení
CAU	Podpora DES, 3DES, AES, MD5, and SHA-1
RNGA	32bitový generátor náhodných čísel
RTC	Poskytuje časové základny s možností přerušení
ACMP	Porovnává dva analogové vstupy
CMT	Výstup pro IR červený signál
IIC1, IIC2	Podporuje sběrnici I2C
KBI	Pin pro externí přerušení
MCG	Víceúčelový generátor hodin
XOSC	Podporuje zdroj hodin z krystalu
CAN	Podporuje sběrnici CAN
SCI1, SCI2	Podporuje sériovou komunikaci např RS-232
SPI1, SPI2	Poskytuje 4pinové sériové komunikace



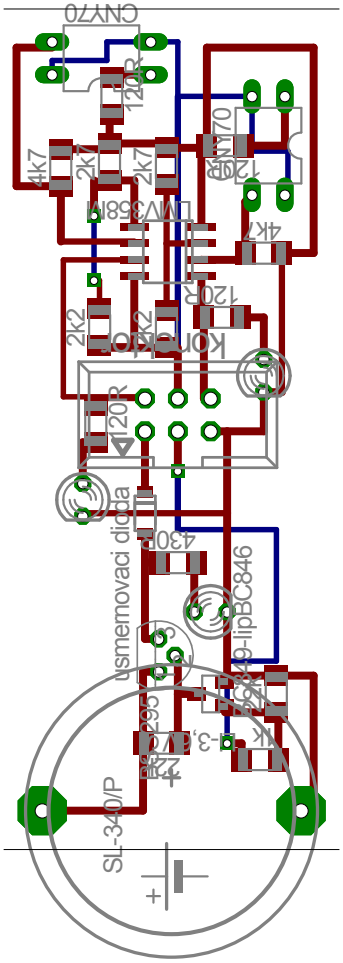
Obrázek 18: Vnitřní blokové schéma mikrokontroléru [1].



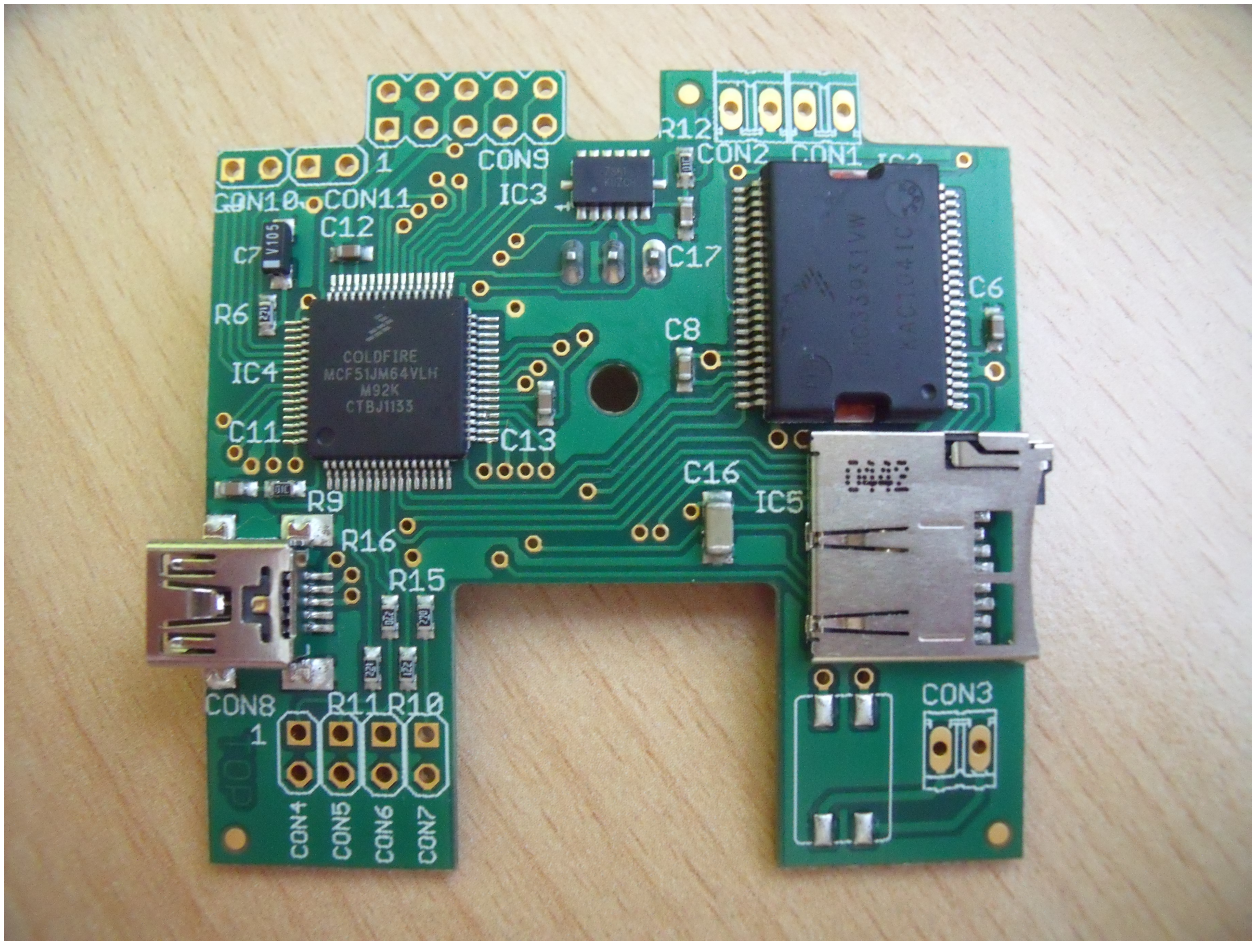
Obrázek 19: Zapojení procesoru.



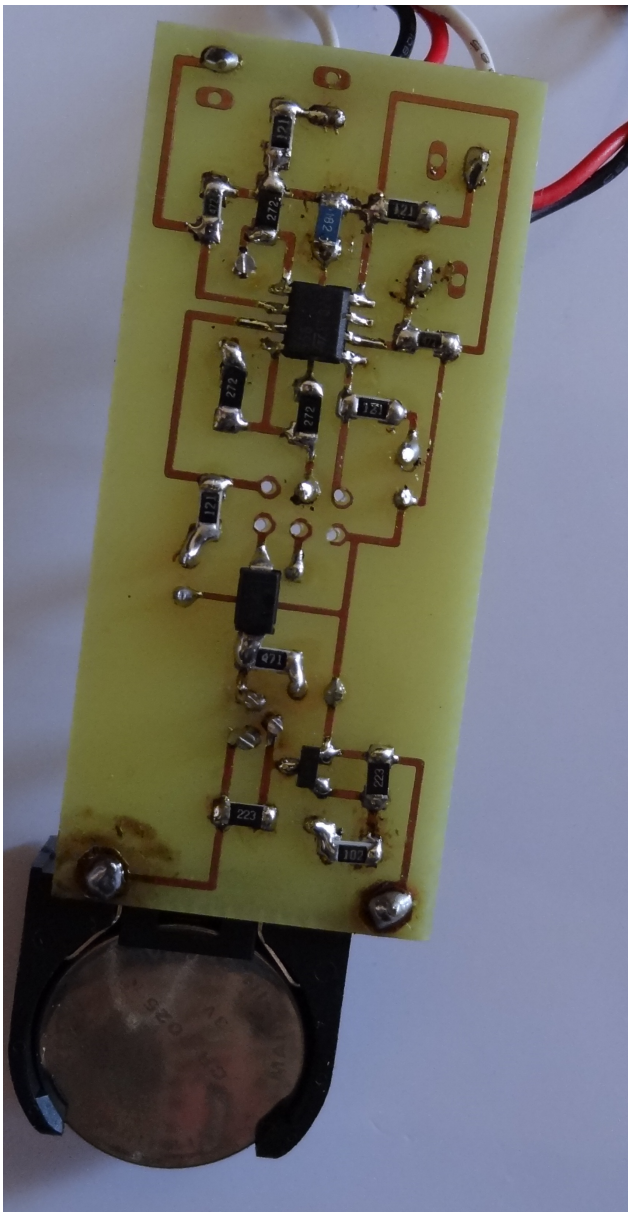
Obrázek 20: Deska základního plošného spoje vytvořena firmou Freescale [10].



Obrázek 21: Deska plošného spoje senzorů vytvořená v prostředí Eagle.



Obrázek 22: Základní plošný spoj dodaný firmou Freescale [10].



Obrázek 23: Deska plošného spoje senzorů.